

Adaptive Policies for Scheduling with Reconfiguration Delay: An End-to-End Solution for All-Optical Data Centers

Chang-Heng Wang, *Student Member, IEEE*, Tara Javidi, *Member, IEEE*,

Abstract—All-optical switching networks have been considered a promising candidate for the next generation data center networks thanks to its scalability in data bandwidth and power efficiency. However, the bufferless nature and the nonzero reconfiguration delay of optical switches remain great challenges in deploying all-optical networks. This paper considers the end-to-end scheduling for all-optical data center networks with no in-network buffer and nonzero reconfiguration delay. A framework is proposed to deal with the nonzero reconfiguration delay. The proposed approach constructs an adaptive variant of any given scheduling policy. It is shown that if a scheduling policy guarantees its schedules to have schedule weights close to the MaxWeight schedule (and thus is throughput optimal in the zero reconfiguration regime), then the throughput optimality is inherited by its adaptive variant (in any nonzero reconfiguration delay regime). As a corollary, a class of adaptive variants of the well known MaxWeight policy is shown to achieve throughput optimality without prior knowledge of the traffic load. Furthermore, through numerical simulations, the simplest such policy, namely the Adaptive MaxWeight (AMW), is shown to exhibit better delay performance than all prior work.

Keywords—Reconfiguration delay, scheduling, throughput optimality

I. INTRODUCTION

Massive data centers serve as the basis of a huge variety of online services and applications nowadays. The underlying network interconnects face increasingly stringent performance requirements such as high data bandwidth and low latency. All-optical networks emerge as a promising candidate for the next generation data center networks and benefit from two technical breakthroughs: (1) the advancement of dense wavelength division multiplexing (DWDM) in optical fibers and (2) the optical switches substituting traditional electronic switches, which typically incur high cost and high power demand when supporting high data bandwidth. Due to the inherently bufferless nature of optical switches, however, the data transmission in an all-optical network will need to be conducted in an end-to-end fashion. This network topology can be viewed as a single crossbar interconnecting the end hosts, except that the full bisection bandwidth is not always guaranteed. In other words, an efficient utilization of the all-optical network depends on efficient centralized controller that can schedule the end-to-end transmissions.

The main challenge of the scheduling for optical networks, as opposed to the traditional electronic crossbar fabric scheduling, comes from the fact that optical networks typically exhibit

a nonzero *reconfiguration delay* upon changing the circuit configuration. This reconfiguration delay comprises of two factors: (1) The reconfiguration delay of the optical switches since candidate technologies (such as binary MEMS [1], ROADM [2]) typically involve mechanically directing laser beams and thus require a certain time to finish a reconfiguration. (2) The time for the control plane to control/communicate with the optical switches along the intended circuit. During the circuit reconfiguration, reliable packet transmission could not be supported in the network. For practical circuit switch technologies, this reconfiguration delay is significantly longer than the link-layer inter-frame gap. For example, the reconfiguration delay for state of the art binary MEMS is $2 - 20 \mu\text{s}$ [1], which is significantly larger than the inter-frame gap of 9.6 ns (for 10 Gigabit Ethernet). This nonzero reconfiguration delay motivates the need for scheduling policies that explicitly account for the reconfiguration delay.

With the presence of the reconfiguration delay, it is well known that the scheduling policy should avoid reconfiguring the circuit too frequently. Most of the existing work for scheduling policies with nonzero reconfiguration delay ([3], [4], [5]) tend to provision the future schedules for a certain time period (usually a long time period). These scheduling policies, classified as “quasi-static” policies, can perform poorly since their schedules may depend on severely out-dated queue length information. In contrast, “dynamic” policies determine each schedule based on the most up-to-date queue length information. An example subclass of dynamic policies is the frame-based policies such as the Fixed-Frame MaxWeight (FFMW) [6], which has good delay performance if the arrival statistics is known in advance. However, the problem for frame-based policies is that the duration of the frame is set in a manner to ensure that the loss of duty cycle due to the reconfiguration delay is negligible relative to the traffic load. In other words, these policies require prior knowledge of the arrival statistics and the value of reconfiguration delay to ensure the stability of the network.

In this paper, we propose a novel class of scheduling policies called adaptive policies for scheduling with nonzero reconfiguration delay. The adaptive policies make scheduling decisions every time slot and determines both the schedule and the time to reconfigure the schedule based on the most recent queue length information. The main idea behind the adaptive policy is to keep the current schedule as long as it is “good enough” so that the schedule reconfiguration only occurs when it is necessary. To be more specific, the construction of an

adaptive policy requires a scheduling policy π and a sublinear function g . At each time slot t , the policy π proposes a schedule $\Pi(t)$, and the adaptive policy computes the schedule weight difference between $\Pi(t)$ and the current schedule. If the weight difference is less than a threshold dependent on the function g , then the current schedule is considered good enough; otherwise the schedule is reconfigured to $\Pi(t)$. The constructed policy is called the g -adaptive variant of π where the function g measures the reluctance of changing the schedule, and is thus referred to as the *hysteresis function*. We show in this paper that if the original policy π has schedule weight that is close to the MaxWeight schedule (either in deterministic or expected sense), which guarantees throughput optimality under a zero reconfiguration delay, then its g -adaptive variant achieves throughput optimality for any fixed reconfiguration delay. Note that this stability guarantee does not require prior knowledge on the arrival statistics or the value of reconfiguration delay to stabilize the network, as opposed to the frame-based policies, such as the FFMW policy.

The rest of the paper is organized as follows. In the next section, the network model and the notion of stability are introduced. In section III, we introduce the class of adaptive policies and analyze its throughput as well as certain delay bounds. We then explain the mechanism of the adaptive policies and compare them with scheduling policies in the literature qualitatively in section IV. Section V gives the performance evaluation and the comparison between scheduling policies through simulations. Finally, we conclude with a summary and some future directions in section VI.

II. SYSTEM MODEL

A. The Network Model

Consider a set of N top of rack (ToR) switches, labeled by $\{1, 2, \dots, N\}$, which are interconnected by an optical switched network, as shown in Fig. 1. Each ToR switch can serve both as a source and a destination simultaneously. We assume no buffering in the optical network, hence all the buffering occurs in the edge of the network, i.e. within the ToR switches. Each ToR switch maintains $N - 1$ edge queues (either physically or virtually), which are denoted by Q_{ij} , where $j \in \{1, 2, \dots, N\} \setminus \{i\}$. Packets going from the ToR switch i to j are enqueued in the edge queue Q_{ij} before transmission.

The system considered is assumed to be time-slotted, with the time indexed as $t \in \mathbb{N}_+ = \{0, 1, 2, \dots\}$. Each slot duration is the transmission time of a single packet, which is assumed to be a fixed value. Let $A_{ij}(t)$ and $D_{ij}(t)$ be the number of packets arrived at and departed from queue Q_{ij} at time t , respectively. Let $L_{ij}(t)$ be the number of packets in the edge queue Q_{ij} at the beginning of the time slot t . For ease of notation, we write $\mathbf{A}(t) = [A_{ij}(t)]$, $\mathbf{D}(t) = [D_{ij}(t)]$, $\mathbf{L}(t) = [L_{ij}(t)]$, where $\mathbf{A}(t), \mathbf{D}(t), \mathbf{L}(t) \in \mathbb{N}_+^{N \times N}$. We adopt the convention that the packet arrivals occur at the end of each time slot, and the queue dynamics is then given by

$$L_{ij}(t+1) = L_{ij}(t) - D_{ij}(t) + A_{ij}(t)$$

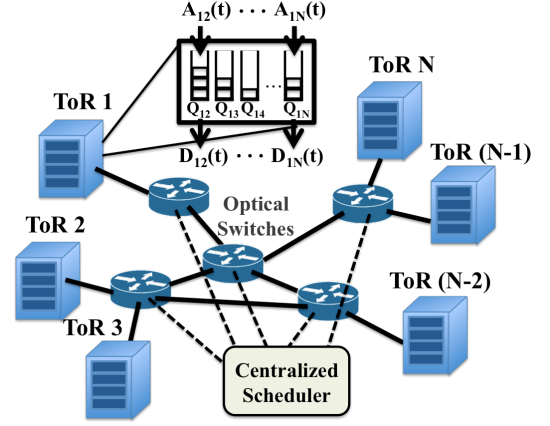


Fig. 1. An example of the system model

We assume the arrival processes $A_{ij}(t)$ to be independent over $i, j \in \{1, 2, \dots, N\}, i \neq j$. Each process $A_{ij}(t)$ is i.i.d. over time slots. We call the mean of $A_{ij}(t)$ as the traffic rate $\lambda_{ij} = \mathbb{E}\{A_{ij}(0)\}$, and define the traffic rate matrix as $\lambda = [\lambda_{ij}] \in \mathbb{R}^{N \times N}$.

B. Schedules and Scheduling Policies

Let $\mathbf{S}(t) \in \{0, 1\}^{N \times N}$ denote the schedule at time t , which indicates the optical circuits established between the ToR switches. We set $S_{ij}(t) = 1$ if an optical circuit from ToR i to ToR j exists at time t , and $S_{ij}(t) = 0$ otherwise. The feasible schedules for the network are determined by the network topology and physical constraints on simultaneous data transmissions. We let \mathcal{S} denote the set of all feasible schedules, i.e. $\mathbf{S}(t) \in \mathcal{S}$ for all t . For instance, it's common to assume at any t each ToR can only transmit to at most one destination, and can only receive from at most one source, i.e. $\sum_i S_{ij}(t) \leq 1, \sum_j S_{ij}(t) \leq 1$. Under such assumption $\mathcal{S} \subset \mathcal{P}$, where \mathcal{P} is the set of all permutation matrices, i.e. if $\mathbf{S}(t)$ has N circuit connections it is a permutation matrix. Note the permutation matrices might not all be feasible in a network; however, if all such schedules are in the feasible schedule set \mathcal{S} , we say the network topology is non-blocking.

Upon reconfiguring a schedule, the network incurs a reconfiguration delay, during which no packet could be transmitted. In this paper we focus on the effect of the reconfiguration delay on the performance of scheduling policies. We make this notion formal through the following two definitions:

Definition 1. Let $\{t_k^S\}_{k=1}^\infty$ denote the time instances when the schedule is reconfigured. The schedule between two schedule reconfiguration time instances remains the same, i.e.

$$\mathbf{S}(\tau) = \mathbf{S}(t_k^S), \quad \forall \tau \in [t_k^S, t_{k+1}^S - 1]$$

Definition 2. Let Δ_r be the reconfiguration delay associated with reconfiguring the schedule of the network. During the period of schedule reconfiguration, no packet transmission could occur in the network. This means that $\forall i, j \in \{1, 2, \dots, N\}$,

$\forall k \in \mathbb{N}_+$, and $0 \leq \tau \leq \Delta_r$, we have $D_{ij}(t_k^S + \tau) = 0$. Note that for all other time, $D_{ij}(t) = S_{ij}(t)$ if $L_{ij}(t) > 0$.

The schedule of the network is determined by a *scheduling policy*, which is defined as below.

Definition 3. A **scheduling policy** determines the schedules $\{\mathbf{S}(t)\}_{t=0}^\infty$ where $\mathbf{S}(t) \in \mathcal{S}$ is measurable with respect to $\sigma\{\{\mathbf{A}(\tau)\}_{\tau=0}^t, \{\mathbf{S}(\tau)\}_{\tau=0}^{t-1}\}$, which is the history up to time t .

Definition 4. A **Markov scheduling policy** is a scheduling policy under which the schedule $\mathbf{S}(t)$ depends on the history solely through the current state $X_t = (\mathbf{S}(t-1), \mathbf{L}(t))$. Restricting attention to this class of policies ensures the process $\{X_t\}_{t=0}^\infty$ to be a Markov process. In this case we denote the Markov scheduling policy as π and use the notation $\mathbf{\Pi}(t)$ to denote the choice of schedule generated by π given the state $X_t = (\mathbf{S}(t-1), \mathbf{L}(t))$.

Notice that we intentionally separate the notation to let $\mathbf{S}(t)$ denote the schedules of the network and $\mathbf{\Pi}(t)$ denote the schedule generated by the Markov policy π .

A specific scheduling policy of interest to our work is the **MaxWeight** policy defined below.

Definition 5. Given a schedule $\mathbf{S} \in \mathcal{S}$, we define the **weight** of schedule \mathbf{S} at time t as

$$W_{\mathbf{S}}(t) = \langle \mathbf{L}(t), \mathbf{S} \rangle = \sum_{i=1}^N \sum_{j=1}^N L_{ij}(t) S_{ij}(t). \quad (1)$$

Definition 6. The **MaxWeight** policy determines the schedule $\mathbf{S}^*(t)$ to be the schedule that has the maximum weight among the feasible schedules at time t , i.e.

$$\mathbf{\Pi}^*(t) = \arg \max_{\mathbf{S} \in \mathcal{S}} W_{\mathbf{S}}(t).$$

We also denote the weight of the MaxWeight schedule at time t as $W^*(t) = \max_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N L_{ij}(t) S_{ij}(t)$ and call it the **maximum weight** at time t .

C. Network Stability

Definition 7. The network is **strongly stable** under policy π or a policy π is said to strongly stabilizes the network if its queue lengths $\mathbf{L}(t)$ satisfies:

$$\limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\|\mathbf{L}(\tau)\|\} < \infty$$

where $\|\mathbf{L}(t)\| = \sum_{i,j=1}^N L_{ij}(t)$ is the total queue length of the system and the expectation is taken with respect to the statistics induced by a random packet arrivals and policy π . This means that a scheduling policy directly controls the statistics of queue occupancies and hence the per packet delay.

With the notion of the strong stability, we may then define the admissible arrival traffic and the throughput optimality of a scheduling policy as follows:

Definition 8. The arrival traffic $\mathbf{A}(t)$ is **admissible** if there exists a scheduling policy which strongly stabilizes the system. The traffic rate matrix $\boldsymbol{\lambda}$ is admissible if $\mathbf{A}(t)$ is admissible.

Definition 9. The **capacity region** is the set of all admissible traffic rate matrices and is denoted as $\bar{\Lambda}$.

We may then define the traffic load as follows:

Definition 10. The **load** of the traffic rate matrix $\boldsymbol{\lambda}$ is defined as

$$\rho(\boldsymbol{\lambda}) = \inf\{r : \boldsymbol{\lambda} \in r\bar{\Lambda}, 0 < r < 1\}$$

where $\bar{\Lambda}$ is the closure of Λ . We shall use ρ instead of $\rho(\boldsymbol{\lambda})$ whenever there is no confusion.

Definition 11. A scheduling policy achieves **throughput optimality** if it strongly stabilizes the network under any admissible arrival traffic.

When $\Delta_r = 0$, several scheduling policies (e.g. [7], [8], [9]) has been shown to achieve throughput optimality in the literature. However, in the regime of $\Delta_r > 0$, these scheduling policies typically loses the throughput optimality guarantee since they do not address the fact that each schedule reconfiguration would result in a significant reduction in the duty cycle and hence decrease the utility of the network. The challenge for scheduling in the $\Delta_r > 0$ regime is that both the quality of the schedules and the rate of schedule reconfiguration affects the performance of a scheduling policy. It is not hard to see that there is a tradeoff between these two factors: Reducing the rate of reconfiguration means that the network is forced to stick with a schedule longer and loses the chance to use a better schedule; on the other hand, pursuing a better schedule most of the time inevitably increases the rate of reconfiguration and thus the incurred overhead. Therefore, a good scheduling policy in the $\Delta_r > 0$ regime must strive to achieve the balance between these two factors.

III. MAIN RESULTS

In this section, we introduce a broad class of adaptive scheduling policies that balance the rate of schedule reconfiguration and the quality of the schedules without prior knowledge of the arrival traffic. We first introduce a general approach that could transform any scheduling policy to an adaptive variant of the original policy and give some example policies generated by this approach. We then introduce several adaptive policies that could achieve throughput optimality in the $\Delta_r > 0$ regime. In particular, we primarily focus on scheduling policies that have the schedule weight close to the MaxWeight policy (either in deterministic sense or in expectation). These scheduling policies have been shown in the literature (e.g. [10]) to achieve throughput optimality when $\Delta_r = 0$. In this paper, we show that under mild conditions, the adaptive variants of these policies achieve throughput optimality under any fixed reconfiguration delay $\Delta_r > 0$.

A. Adaptive Policies

Given a Markov scheduling policy π and current state X_t , let $\mathbf{\Pi}(t) = \pi(X_t)$ denote the schedule generated by π at time t . We then define the following:

- $W^\pi(t) = W_{\Pi(t)}(t)$ is the weight of the schedule $\Pi(t)$ at time t
- $\tilde{W}(t) = W_{\mathbf{S}(t-1)}(t)$ is the weight of the previous schedule $\mathbf{S}(t-1)$ at time t

Now note that at any given time t , $\Delta W(t) = W^\pi(t) - \tilde{W}(t)$ measures the potential improvement (in terms of schedule weight (1)) associated with following policy π instead of sticking with the previously used schedule. Since each schedule change results in a loss in duty cycle, our proposed class of adaptive policies show some inertia against frequent schedule reconfiguration. More precisely, let us define a **hysteresis function** g where $g : \mathbb{R} \rightarrow \mathbb{R}$ is a nonnegative, continuous, strictly increasing, and sublinear (i.e. $\lim_{x \rightarrow \infty} \frac{g(x)}{x} = 0$) function.

Our proposed g -adaptive Markov policy π^g uses the new schedule $\Pi(t)$ only if $\Delta W(t) > g(W^\pi(t))$. In other words,

$$\begin{aligned} \Pi^g(t) &= \pi^g(\mathbf{S}(t-1), \mathbf{L}(t)) \\ &= \begin{cases} \mathbf{S}(t-1) & \text{if } \Delta W(t) \leq g(W^\pi(t)) \\ \Pi(t) & \text{if } \Delta W(t) > g(W^\pi(t)) \end{cases} \end{aligned}$$

Note that the proposed g -adaptive Markov policy could be constructed from any Markov scheduling policy. Given the Markov policy π , we call π^g the g -adaptive variant of π .

The intuition behind this construction is that the g -adaptive policy holds on to the last schedule as long as it is “good enough” relative to the current schedule generated by π , $\Pi(t)$. The sublinearity of the function $g(\cdot)$ is a technical assumption used to achieve the throughput optimality, which would become clear in the analysis given in the next subsection.

We now give some examples for possible combinations of the function $g(\cdot)$ and the scheduling policy π .

Example 1. (the Adaptive MaxWeight policy [11])

Let the scheduling policy π be the MaxWeight policy, and the function $g(\cdot)$ takes the form $g(x) = (1 - \gamma)x^{1-\delta}$, where $\gamma \in (0, 1)$, $\delta \in (0, 1)$. Then the g -adaptive variant of π is called the Adaptive MaxWeight, as introduced in [11].

The Adaptive MaxWeight policy computes the weight difference between the MaxWeight schedule and its last schedule $\Delta W = W^*(t) - \tilde{W}^*(t)$ to the threshold $g(W^*(t)) = (1 - \gamma)(W^*(t))^{1-\delta}$ at each time slot. It reconfigures to the MaxWeight schedule if the difference is above the threshold, and keeps the current schedule if otherwise.

Example 2. (the g -adaptive variant of the Pipelined MaxWeight policy)

Under the pipelined MaxWeight policy [12], the scheduler determines the schedule at time t to be the schedule maximizing the weight at time $t - K$, for some fixed scalar $K < \infty$. Intuitively, one can think of the pipelined MaxWeight as a scheduler that initiates MaxWeight computation at each time slot but obtains and enforces it only K time slots later. Therefore, the schedule at time t is the MaxWeight schedule based on the queue length information at time $t - K$, i.e. $\mathbf{L}(t - K)$. The selection of the function $g(\cdot)$ could be any continuous, strictly increasing, and sublinear function, e.g. $g(x) = (1 - \gamma)x^{1-\delta}$ as in Example 1.

Since the MaxWeight scheduling policy is well known for its high computation complexity, in the literature several lower complexity policies have also been proposed with good stability conditions when $\Delta_r = 0$. We may consider the adaptive variant of these policies as well.

Example 3. (the g -adaptive variant of the Tassiulas random policy)

The Tassiulas Random policy [8] utilizes random schedule selection and memory to determine the schedule. It compares the weight between the last schedule and a randomly selected schedule (according to an arbitrary distribution on the feasible schedule \mathcal{S} , say uniformly random). Let $\mathbf{Z}(t)$ be the randomly selected schedule at time t , then the schedule determined by the Tassiulas random policy is given by

$$\Pi_T(t) = \begin{cases} \Pi_T(t-1) & \text{if } W_{\Pi_T(t-1)}(t) \geq W_{\mathbf{Z}(t)}(t) \\ \mathbf{Z}(t) & \text{otherwise} \end{cases}$$

Example 4. (the g -adaptive variant of the Hamiltonian policy)

The Hamiltonian policy [9] utilizes the Hamiltonian walk on the set of permutation matrices and memory to determine the schedule. It compares the schedule weight between the last schedule and the schedule on the Hamiltonian path and select the schedule with higher weight. Specifically, let $\mathbf{H}(t)$ be the schedule on the Hamiltonian path at time t , then the schedule determined by the Hamiltonian policy is given by

$$\Pi_H(t) = \begin{cases} \Pi_H(t-1) & \text{if } W_{\Pi_H(t-1)}(t) \geq W_{\mathbf{H}(t)}(t) \\ \mathbf{H}(t) & \text{otherwise} \end{cases}$$

Example 5. (the g -adaptive variant of the Maximum Size scheduling policy)

A maximum size scheduling policy [13] selects a schedule that has the maximum number of nonempty queues. It can be viewed as a variation of the MaxWeight policy except that each nonempty queue has weight one, and each empty queue has weight zero. In particular, the schedule is selected as

$$\Pi_{MS}(t) = \arg \max_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N S_{ij}(t) \mathbb{1}_{\{L_{ij}(t) > 0\}}$$

where $\mathbb{1}_{\{L_{ij}(t) > 0\}}$ is the indicator function for the event $\{L_{ij}(t) > 0\}$.

In the next subsection we discuss conditions required for an adaptive variant to achieve throughput optimality under $\Delta_r > 0$. Notice that the maximum size policy does not achieve throughput optimality even in the $\Delta_r = 0$ regime [13], therefore we do not consider the throughput analysis of example 5.

B. Drift Analysis

Throughout this subsection, we impose the following assumption on the arrival traffic:

Assumption 1. Assume the arrival at each queue is bounded from above, i.e. $\mathbf{A}_{ij}(t) \leq A_{max} < \infty, \forall i, j \in \{1, 2, \dots, N\}, \forall t$.

We start with a class of scheduling policies that guarantee bounded schedule weight difference to the MaxWeight schedule at all times.

Condition 1. Suppose there exists a constant $G < \infty$ such that the schedule weight of the scheduling policy π , $W^\pi(t) = W_{\Pi(t)}(t)$, satisfies

$$W^\pi(t) \geq W^*(t) - G, \quad \forall t$$

where $W^*(t)$ is the maximum weight, the schedule weight of the MaxWeight schedule at time t as introduced in Section II.

Given a scheduling policy π that satisfies Condition 1, the following Theorem establishes that the g -adaptive variant of π achieves the throughput optimality.

Theorem 1. Given any reconfiguration delay $\Delta_r > 0$. Assume the traffic is admissible and satisfies assumption 1. Suppose a scheduling policy π satisfies Condition 1, then the g -adaptive variant of π is throughput optimal.

Moreover, assume further that $g(\cdot)$ is concave. Let $G < \infty$ be the constant in condition 1. Let $M = g^{-1}(G + N(A_{\max} + 1)T) + NT$, then the g -adaptive variant of π guarantees the mean queue length to satisfy

$$\lim_{t \rightarrow \infty} \mathbb{E}[\|\mathbf{L}(t)\|] \leq \inf_{T > \frac{\Delta_r}{1-\rho}} \tilde{L}_T \quad (2)$$

where \tilde{L}_T is the fixed point solution to the equation:

$$\tilde{L}_T = \frac{N}{1 - \frac{\Delta_r}{T} - \rho} \left\{ g(\tilde{L}_T + NA_{\max}T) + G + N(T + A_{\max}\Delta_r) + M(1 + NA_{\max}) + \frac{N^2 A_{\max}^2}{2} \right\}$$

The proof of Theorem 1 is given in Appendix. The proof utilizes the Foster-Lyapunov Theorem and consists of two main components. The first one shows that the schedule of the adaptive policy π^g has weight difference to the MaxWeight schedule bounded by a sublinear function of the maximum weight. This potentially gives the negative Lyapunov drift, as similarly argued in [12]. The second part is that the rate of schedule reconfiguration becomes smaller as the queue lengths become larger. With this property, we show that the overhead incurred by the schedule reconfiguration delay becomes arbitrarily small when the total queue lengths $\|\mathbf{L}(t)\|$ increases. This suffices to give the guarantee of negative expected drift when $\|\mathbf{L}(t)\|$ is large and thus guarantee the stability.

With Theorem 1, we are now ready to show the throughput optimality of some example adaptive policies given in the previous subsection.

Corollary 1. The adaptive policies in examples 1 and 2 achieve throughput optimality.

Proof: For example 1, since π is the MaxWeight policy, Condition 1 is satisfied by definition, with $G = 0$. As for example 2, Condition 1 is satisfied with $G = N(A_{\max} + 1)K$. To see this, note that at most one packet could depart from

each queue at each time slot, and hence

$$\begin{aligned} \langle \mathbf{L}(t), \Pi(t) \rangle &\geq \langle \mathbf{L}(t-K), \Pi(t) \rangle - NK \\ &= \langle \mathbf{L}(t-K), \Pi^*(t-K) \rangle - NK. \end{aligned} \quad (3)$$

Since the arrivals are bounded by A_{\max} , we also have

$$\langle \mathbf{L}(t), \Pi^*(t) \rangle \leq \langle \mathbf{L}(t-K), \Pi^*(t-K) \rangle + NA_{\max}K. \quad (4)$$

From (3) and (4), we have $\langle \mathbf{L}(t), \Pi(t) \rangle \geq \langle \mathbf{L}(t), \Pi^*(t) \rangle - N(A_{\max} + 1)K$ and thus Condition 1 is satisfied. ■

Corollary 2. The g -adaptive variant of the Hamiltonian policy given in example 4 achieves throughput optimality.

Proof: In [9] the Hamiltonian policy is shown to satisfy Condition 1 with $G = 2N(N!)$, hence by Theorem 1 it achieves throughput optimality. ■

As we saw in example 3, some low complexity scheduling policies utilize the random schedule selection and memory to approximate the MaxWeight policy (e.g. [8], [9]). For a random scheduling policy π , the schedule $\Pi(t)$ is a sequence of random schedules from the feasible schedule set \mathcal{S} . These policies guarantee bounded weight differences to the MaxWeight schedule in the expected sense:

Condition 2. Consider a Markov scheduling policy π with weight $W^\pi(t)$ that satisfies the property:

$$\mathbb{E}^\pi[W^\pi(t)|\mathbf{L}(t)] \geq W^*(t) - G, \quad \forall t \quad (5)$$

where $G < \infty$ is a constant, and the expectation \mathbb{E}^π is taken with respect to the policy π .

We consider the throughput optimality of the adaptive variants of policies satisfying the above condition as well.

Theorem 2. Given any reconfiguration delay $\Delta_r > 0$. Assume the traffic is admissible and satisfies assumption 1. For any scheduling policy π that satisfies Condition 2, the g -adaptive version of π achieves throughput optimality.

With Theorem 2, we can extend Corollaries 1 and 2 to the g -adaptive variant of the Tassiulas random policy in example 3:

Corollary 3. The g -adaptive variant of the Tassiulas random policy given in example 3 achieves throughput optimality.

Proof: Let $\Pr\{\mathbf{Z}(t) = \Pi^*(t)\} \geq \epsilon > 0$ for all t and for some ϵ . In [14] the Tassiulas random policy is shown to satisfy Condition 2 with $G = 2N\frac{1-\epsilon}{\epsilon}$, hence by Theorem 2 it achieves throughput optimality. ■

C. Generalization

In the previous subsection, we established the throughput optimality of the g -adaptive variant of any scheduling policy that has schedule weight close to the MaxWeight policy, either in deterministic or expected sense. In the literature the idea of MaxWeight policy has been generalized to a broader class of MaxWeight- f policy by extending the definition of schedule

weight to a more general f -weight. This extended class of policies have been shown to achieve throughput optimality. In this subsection we briefly discuss the criteria for the adaptive variants of these policies to achieve throughput optimality.

We first introduce the MaxWeight- f policy. Given a function $f : \mathbb{R} \rightarrow \mathbb{R}$ which is strictly increasing, and $f(0) = 0$, we define the f -weight of a schedule $\mathbf{S} \in \mathcal{S}$ at time t as

$$W_{f,\mathbf{S}}(t) = \langle f(\mathbf{L}(t)), \mathbf{S} \rangle = \sum_{i=1}^N \sum_{j=1}^N f(L_{ij}(t)) S_{ij}(t).$$

The MaxWeight- f policy is a generalization of the MaxWeight policy in that it selects the schedule $\mathbf{\Pi}_f^*(t)$ that has the maximum f -weight among the feasible schedules, i.e.

$$\mathbf{\Pi}_f^*(t) = \arg \max_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N f(L_{ij}(t)) S_{ij}(t).$$

We denote the f -weight of the MaxWeight- f schedule at time t as $W_f^*(t) = \arg \max_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N f(L_{ij}(t)) S_{ij}(t)$.

Recall that we construct the g -adaptive policies using schedule weight comparison, here we generalize the approach by using the more general f -weight comparison, and refer to it as the (g, f) -adaptive policies. Formally, given a Markov scheduling policy π and current state X_t , let $\mathbf{\Pi}(t) = \pi(X_t)$ denote the schedule generated by π at time t . Let $f(\cdot)$ be the weight function. We then define the following:

- $W_f^\pi(t) = W_{f,\mathbf{\Pi}(t)}(t)$ is the f -weight of the schedule $\mathbf{\Pi}(t)$ at time t
- $\tilde{W}_f(t) = W_{f,\mathbf{S}(t-1)}(t)$ is the f -weight of the previous schedule $\mathbf{S}(t-1)$ at time t

We then define the (g, f) -adaptive variant of π , denoted as $\pi^{g,f}$, with the schedule $\mathbf{\Pi}^{g,f}(t)$ determined as follows:

$$\mathbf{\Pi}^{g,f}(t) = \begin{cases} \mathbf{S}(t-1) & \text{if } \Delta W_f(t) \leq g(W_f^\pi(t)) \\ \mathbf{\Pi}(t) & \text{if } \Delta W_f(t) > g(W_f^\pi(t)) \end{cases} \quad (6)$$

where $\Delta W_f(t) = W_f^\pi(t) - \tilde{W}_f(t)$ is the f -weight difference between the schedule $\mathbf{\Pi}(t)$ and the previous schedule $\mathbf{S}(t-1)$.

An example of the (g, f) -adaptive policies is given below.

Example 6. (the (g, f) -adaptive MaxWeight- α policy, $\alpha > 0$)

For any fixed $\alpha > 0$, let the weight function be $f(x) = x^\alpha$, then the MaxWeight- f policy is also referred as MaxWeight- α policy in the literature. Under the MaxWeight- α policy, the schedule at time t is given by

$$\mathbf{\Pi}_{MW\alpha}(t) = \arg \max_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N L_{ij}^\alpha(t) S_{ij}(t)$$

Recall the hysteresis function $g(\cdot)$ is a strictly increasing and sublinear function. While to ensure the throughput optimality, further restrictions on $g(\cdot)$ would be required, which may be dependent on the weight function $f(\cdot)$. This is shown in the following proposition.

Proposition 1. *Given the weight function $f(x) = x^\alpha$ with $\alpha > 0$. Suppose the sublinear function $g(\cdot)$ satisfies $\lim_{x \rightarrow \infty} \frac{x^{\alpha-1}}{g(x^\alpha)} = 0$,*

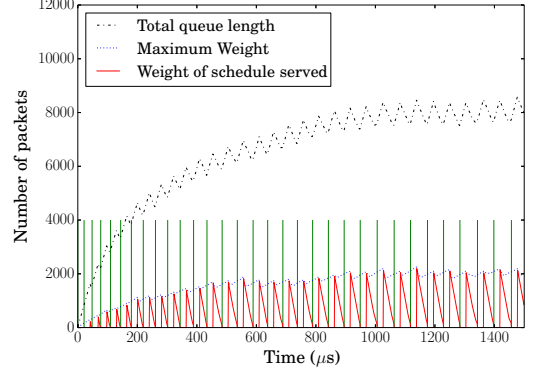


Fig. 2. Sample paths of the total queue length, maximum weight, and the schedule weight, under the Adaptive MaxWeight (AMW) policy. The number of ToRs is $N = 8$, the traffic load is $\rho = 0.6$, and the parameters of the AMW policy are $\delta = 0.01, \gamma = 0.1$. The green lines mark the schedule reconfiguration time instances t_k^S .

then the (g, f) -adaptive variant of the MaxWeight- α in example 6 achieves throughput optimality.

The proof requires modifying the proof of Theorem 1 by taking the Lyapunov function as $V(\mathbf{L}) = \sum_{i,j=1}^N F(L_{ij})$ where $F(x) = \int_0^x f(y)dy$, and is omitted here for brevity.

IV. DISCUSSION

In this section, we first give a brief explanation on how the adaptive policies achieve the throughput optimality. We then categorize policies dealing with nonzero reconfiguration delay into two classes, namely quasi-static and dynamic policies.

A. Queue Dynamics under the Adaptive MaxWeight Policy

Fig. 2 illustrates the sample paths of the total queue lengths, the maximum weight, and the schedule weight of a network under the Adaptive MaxWeight policy. The green vertical lines mark the times of schedule reconfiguration, t_k^S , and the schedule weight is set as zero during the reconfiguration delay (time period of length Δ_r following each schedule reconfiguration instance t_k^S). We first observe that the schedule duration increases as the total queue length increases. This is an essential component in achieving the network stability: let T be the mean schedule duration, then in order to ensure stability under $\Delta_r > 0$, the rate of schedule reconfiguration must satisfy $1 - \frac{\Delta_r}{T} > \rho$. The schedule duration increases with the queue length until this condition is satisfied.

A somewhat more interesting observation is in the mechanism of this schedule duration adjustment. Recall that in the construction of (g, f) -adaptive policy, it requires no explicit adjustment of the schedule duration, i.e. the duration of a schedule is not explicitly set at the time it is reconfigured. Instead, through the schedule determination by weight comparison at each time slot, the schedule duration is implicitly “adapted” to the appropriate value. After each schedule reconfiguration, the schedule weight begins from the

maximum weight and decreases as the network is serving the queues associated with the schedule. When the total queue length is larger (and thus the maximum weight is larger since $W^*(t) \geq \frac{1}{N} \|\mathbf{L}(t)\|$), the threshold is larger and it takes longer for the weight difference between the maximum weight and the schedule weight to surpass the threshold. This property may be characterized by lemma 1 shown in Appendix A.

Note that this mechanism differs from other scheduling policies in the literature that explicitly adjust the schedule duration ([3], [4], [5]). We give a brief introduction of these policies and categorize them based on this schedule provisioning behavior in the next subsection. The performance comparison of these policies is then evaluated through simulations and presented in the next section.

B. Quasi-static v.s. Dynamic Policies

For scheduling policies accounting for nonzero reconfiguration delay, we classify them into two categories: “quasi-static scheduling” and “dynamic scheduling.” Quasi-static scheduling policies, also referred to as “batch scheduling,” [3] select a series of schedules based on a single schedule computation process, as shown in Figure 3 (a). We argue that under these policies, the generated schedules may depend on very out-dated information, especially for schedules employed later in a batch. This is the source of significant performance degradation. In contrast, under dynamic scheduling policies, each schedule is generated based on the most up-to-date queue information, as shown in Figure 3 (b). We now describe several example policies for each class and discuss their performance.

In [3], batch scheduling policies are further classified as fixed batch scheduling (FBS) policies or adaptive batch scheduling (ABS) policies depending on how to determine the time instances for schedule computation. If the time between two schedule computation is a fixed duration then it is a FBS policy, otherwise it is a ABS policy. In general, a FBS policy does not guarantee stability unless the number of schedules employed within a batch is restricted to avoid too frequent schedule reconfiguration. The traffic matrix scheduling (TMS) policy [4] is a special case of FBS policy which could guarantee stability if the traffic load is known in advance. Under the TMS policy, the schedule computation occurs at $t_k = kW$ for some integer parameter $W < \infty$ and $k = 0, 1, \dots$. The TMS policy utilizes the Birkhoff von-Neumann (BvN) decomposition [15] to determine the schedules in the following W time slots.¹ More specifically, the scheduler takes the queue length information $\mathbf{L}(t_k)$ and scales it to a doubly stochastic matrix $\mathbf{B}(t_k)$ which indicates the relative service requirement in the following W slots and performs the BvN decomposition on $\mathbf{B}(t_k)$ as $\mathbf{B}(t_k) = \sum_{i=1}^Q \alpha_i \mathbf{P}_i$. Note the number of terms Q in the decomposition may vary (while $Q \leq N^2 - 2N + 2$). In practice, the parameter Q is set as a fixed number to avoid excessive schedule changes and Q largest weighted schedules are chosen. If the arrival traffic load is known a priori, an appropriate choice of parameters ensures the stability of TMS.

¹The BvN decomposition is based on the BvN Theorem [16] that every doubly stochastic matrix could be decomposed as a convex combination of permutation matrices

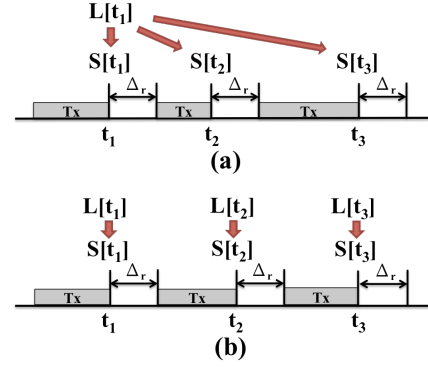


Fig. 3. Timing diagram for different scheduling strategies. (a) Quasi-static policies: Series of schedules determined in a single schedule computation, and some schedules could depend on out-dated queue information when being deployed. (b) Dynamic policies: Each schedule is computed based on the most up-to-date edge queue information.

The ABS policy proposed in [3] determines each schedule computation time as the time the packets from last batch are cleared. The ABS guarantees rate stability [3] which is a weaker notion of stability compared to the strong stability considered in this paper. In fact, the expected queue length may be unbounded under the ABS policy, hence it is not considered in the performance comparison in this work. On the other hand, the Variable Frame MaxWeight (VFMW) policy proposed in [5] can be viewed as a variant to the ABS policy. The VFMW policy selects only one schedule for each batch, and the batch duration is a function of the batch size (instead of being the time that the batch is cleared). While the VFMW policy is shown in [5] to be throughput optimal, it has poor delay performance since it selects and fixes the schedule duration disregarding the arrivals in the schedule duration, which is a similar problem to other quasi-static policies.

In contrast to the quasi-static policies, under the dynamic scheduling policies, the schedule being employed at each schedule reconfiguration instance is based on the most up-to-date queue length information. Frame-based policies, such as the Fixed Frame MaxWeight (FFMW) policy [6], are examples for dynamic policies. The FFMW policy selects a fixed period T and sets the schedule reconfiguration times at $t_n = nT, n = 0, 1, 2, \dots$. At each t_n , the schedule is selected as the MaxWeight schedule at time t_n , therefore each schedule is based on the most up-to-date queue information and result in an improved delay performance. Unfortunately, like the TMS policy, however, the FFMW requires prior knowledge of the traffic load to guarantee the network stability which is similar to the TMS policy. Note that by the definition given, our proposed (g, f) -adaptive policies are other examples of dynamic policies. In contrast to the FFMW, however, our proposed (g, f) -adaptive policies achieve throughput optimality without prior knowledge of the traffic load. Moreover, the (g, f) -adaptive policies also have good delay performance, as would be shown through simulations in the next section.

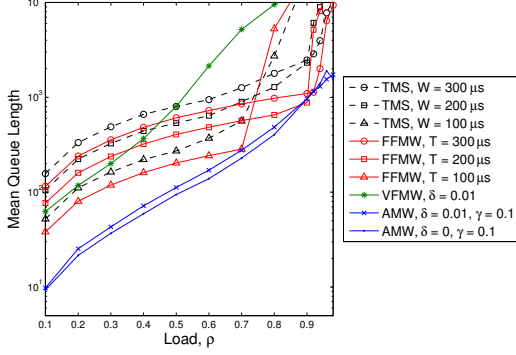


Fig. 4. Mean queue length versus traffic load ρ under the **uniform traffic**. The TMS policy reconfigures the schedule $q = 10$ times within qT time duration. The scheduling rate under either the TMS or PMW is equal to $1/T$, while under AMW is adapted to the traffic load intensity.

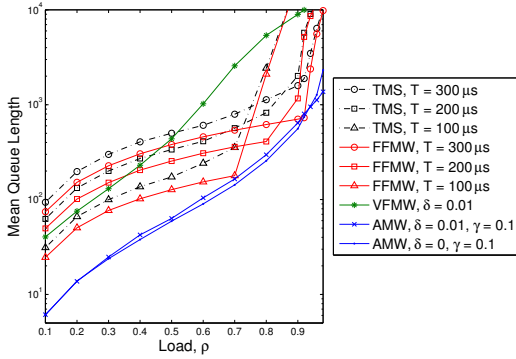


Fig. 5. Mean queue length versus traffic load ρ under the **nonuniform traffic**. The TMS policy reconfigures the schedule $q = 10$ times within qT time duration.

V. SIMULATION

In this section we present simulation results for the AMW policy in Example 1, and compare them to the benchmark scheduling policies such as TMS, FFMW and VFMW. We also present comparison of the AMW policy against adaptive variants of lower complexity policies approximating the MaxWeight policy, as given in the examples in section III-A.

The experiments are conducted with the simulator built for the REACToR switch in [17]. The reconfiguration delay is $\Delta_r = 20 \mu s$. In order to compare scheduling policies in optical switches, we cease the electronic switches in the hybrid switch in [17] and only utilize the optical switches. We consider $N = 100$ ToR switches, and the network topology is assumed to be non-blocking. Therefore, the set of feasible schedules \mathcal{S} is in fact the set of $N \times N$ permutation matrices. Each link has data bandwidth $B = 100$ Gbps, and the packets are of the same size $p = 1500$ bytes (each takes $0.12 \mu s$ for transmission). Each edge queue can store up to 1×10^5 packets, and incoming packets are discarded when the queue is full.

The traffic is assumed to be admissible, i.e. $\rho(\lambda) < 1$, while the load matrices λ are classified as the following types:

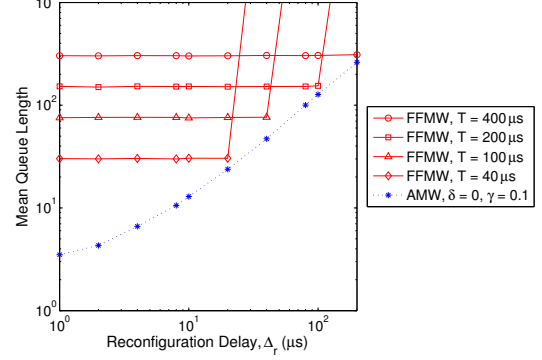


Fig. 6. Mean queue length versus the reconfiguration delay Δ_r under the nonuniform traffic. The traffic load is fixed as $\rho = 0.3$.

- 1) Uniform: $\lambda_{ij} = \rho/N, \forall 1 \leq i, j \leq N$.
- 2) Nonuniform: $\lambda_{ij} = \frac{\rho}{M} \sum_{m=1}^M \mathbf{P}_{ij}^m$ where $\mathbf{P}^m, m = 1, \dots, M \in \mathcal{P}$ are permutation matrices picked at random. The number M determines the skewness of the load matrix. We set $M = 100$ here.

The performance measure used is the mean edge queue length (averaged over queues and over time). Notice that the expected average delay of the entire network is linearly related to this quantity according to the Little's law.

In Figs 4 and 5, we compare the scheduling policies described in section IV under the uniform and the nonuniform traffic, respectively. For TMS, we set the number of schedules used between two schedule computation time instances to be $Q = 10$. In Figs. 4 and 5 we can see that the TMS and FFMW perform comparably with the FFMW slightly outperforming the TMS under the same schedule reconfiguration rate $1/T = 10/W$. We note that under both the TMS and FFMW policies, the traffic loads they could stabilize are determined by the reconfiguration rate $1/T = 10/W$. In general, a smaller T (W) value gives better delay performance at a fixed stable load, but choosing a smaller T (W) value also decreases the maximum load that the TMS or FFMW policy could stabilize. We also note that the VFMW policy, which is shown to achieve the throughput optimality without requiring the knowledge of the arrival traffic, in practice, underperforms both the FFMW and TMS policies. On the other hand, the AMW policy ensures an improved performance over the FFMW and TMS policies and achieves throughput optimality.

We now consider the effect of the reconfiguration delay Δ_r to the performance. In Fig. 6, we show the performance of the FFMW and AMW under various Δ_r , while the traffic load is fixed as $\rho = 0.3$. We can see that the AMW outperforms the FFMW under each Δ_r value, regardless of the parameter selection of the FFMW policy. Although there exists an optimal schedule reconfiguration period T of the FFMW policy that achieves comparable performance to the AMW policy at each Δ_r , the choice of the optimal T is dependent on the traffic load ρ . We can see that the performance of the AMW actually traces the optimal performance of the FFMW. This

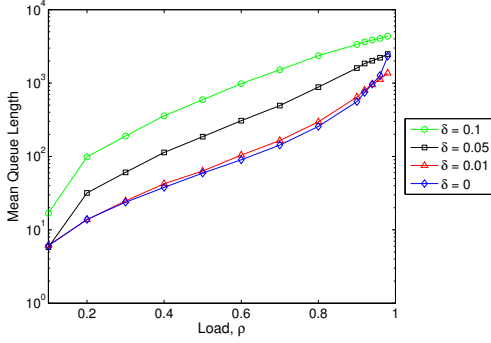


Fig. 7. Mean queue length versus traffic load for the AMW policy under sublinear exponent $\delta \in \{0, 0.01, 0.05, 0.1\}$. The threshold ratio is fixed as $\gamma = 0.1$.

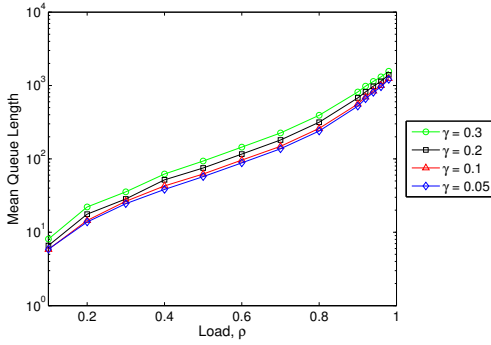


Fig. 8. Mean queue length versus traffic load for the AMW policy under threshold ratio $\gamma \in \{0.05, 0.1, 0.2, 0.3\}$. The sublinear exponent is fixed as $\delta = 0.01$.

observation suggests that the adaptive strategy of the AMW in fact allows it to capture the optimal schedule reconfiguration rate based solely on the queue lengths information and no prior knowledge of the arrival statistics is required.

In Figs. 7 and 8 we consider the effect of the parameter selection for the AMW policy. Recall from example 1 that the g function is selected as $g(x) = (1 - \gamma)x^{1-\delta}$. Fig. 7 shows the performance for sublinear exponent $\delta \in \{0, 0.01, 0.05, 0.1\}$, while the ratio threshold is fixed as $\gamma = 0.1$. Note that at any fixed traffic load ρ , the mean queue length becomes shorter when δ is smaller. Since our analysis (Corollary 1) requires $\delta > 0$, we select $\delta = 0.01$ for the remaining simulations. Fig. 8 presents the performance under the ratio threshold $\gamma \in \{0.05, 0.1, 0.2, 0.3\}$, while the sublinear exponent is fixed as $\delta = 0.01$. Notice that the mean queue length decreases as γ decreases, however, this sensitivity to the variable γ seems to be fairly insignificant.

One of the shortcomings of the AMW policy is the complexity of computing the MaxWeight schedules. Next, we consider the adaptive variants of lower complexity scheduling policies introduced in Examples 3-5. In Fig. 9 we show the mean queue length versus traffic load for the Adaptive Hamiltonian (AHam), the Adaptive Tassiulas random policy (ATass), the

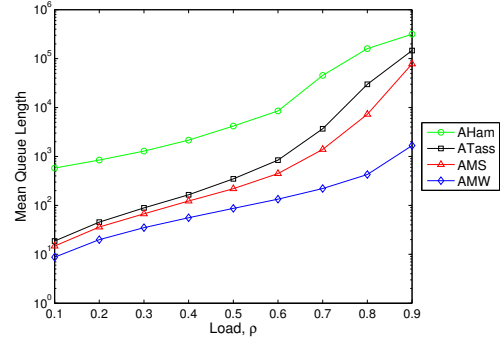


Fig. 9. Mean queue length versus traffic load for different adaptive policies. The number of ToRs is $N = 8$ and $g(x) = (1 - \gamma)x^{1-\delta}$ where $\gamma = 0.1, \delta = 0.01$.

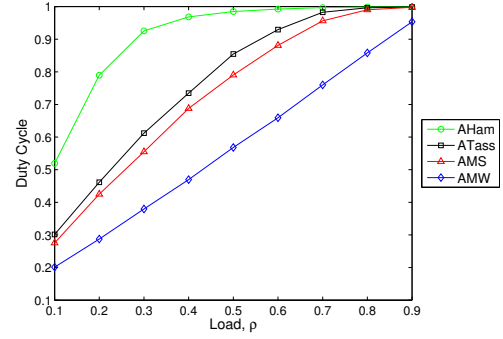


Fig. 10. Duty cycle versus traffic load for different adaptive policies. The number of ToRs is $N = 8$ and $g(x) = (1 - \gamma)x^{1-\delta}$ where $\gamma = 0.1, \delta = 0.01$.

Adaptive Maximum Size (AMS), and the AMW policies, under uniform traffic. Note that the delay performance of the lower complexity policies degrade drastically at large number of ToRs ($G \approx N!$ in Condition 1 for AHam and in Condition 2 for ATass), hence we set $N = 8$ for the simulations here. We also show the duty cycle of these policies in Fig. 10, where the duty cycle is defined as $DC \triangleq 1 - \frac{\Delta_r}{\mathbb{E}\{T\}}$, while $\mathbb{E}\{T\}$ is the mean schedule duration. Note that a necessary condition for a scheduling policy to be throughput optimal under reconfiguration delay $\Delta_r > 0$ and traffic load ρ , is to satisfy $DC > \rho$, which is satisfied by all the scheduling policies shown here. Interestingly, due to the uniformity of the traffic, the AMS policy stabilizes the network and has comparable delay performance to the ATass policy, despite the fact that the AMS policy is not throughput optimal in general.

From Fig. 9, we could see that the delay performance for the lower complexity policies are worse than the AMW policy. We may also observe that as the traffic load gets larger, the performance difference increases. This is because the schedule weight decreases in a slower rate (due to higher arrival rate), and it takes more time for the lower complexity policies to find a schedule with high enough weight. We could see in Fig. 10 that the schedule durations become significantly long

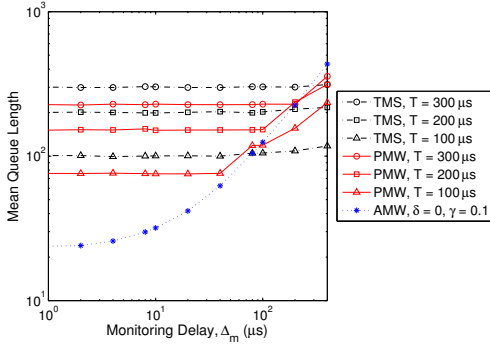


Fig. 11. Mean queue length versus monitoring delay Δ_m under nonuniform traffic. The traffic load is fixed as $\rho = 0.3$. The edge queue state is monitored/updated every microsecond, $t_{k+1}^M - t_k^M = 1\mu s$ for all k .

($DC \rightarrow 1$) at high traffic loads. We also see that the AHam policy has the worst delay performance for all traffic loads, this is because the Hamiltonian walk only changes served queues in a time slot, and takes longer to find the next good schedule.

VI. CONCLUSION

This paper considers the end-to-end scheduling problem in all-optical data center networks. The entire network is viewed as a generalized crossbar interconnect with nonzero schedule reconfiguration delay. Due to the schedule reconfiguration delay, many throughput optimal scheduling policies (under zero reconfiguration delay) in the literature could not be directly applied in this problem.

In this paper, we propose a general method to develop a class of scheduling policies for scheduling with nonzero reconfiguration delay, namely the (g, f) -adaptive variant policies: Given any Markov policy π , a weight function $f(\cdot)$, and a sublinear hysteresis function $g(\cdot)$, we construct a (g, f) -adaptive variant of π which involves a weight comparison between the current schedule and the schedule generated by π , and reconfigure to the schedule generated by π when it is “significantly better.” We show the throughput optimality of the (g, f) -adaptive variants of π given the weight of schedule generated by π is guaranteed to be close enough to the maximum weight (either in the deterministic or the expected sense).

The proposed scheduling policies consider zero in-network buffer mainly due to the inherently bufferless nature of the optical circuits. It is interesting to note that the notion of edge-buffering and end-to-end scheduling has also been explored in the regime of electronic packet-switched data center network [18] recently, in an effort to reduce buffering and congestion within the network. This suggests that our proposed scheduling policies can also be utilized in the context of electronic packet switches (with in-network buffering) in order to further reduce delay and improve performance.

In this work, we consider primarily the effect of the reconfiguration delay to the scheduling policies. In practice, however, monitoring (collecting queue length information) and schedule computation may also take a non-negligible time (denoted as Δ_m and Δ_c , respectively), as suggested in [11]. These

delays can be viewed as incurring delay on the queue length information as in the Pipelined MaxWeight in example 2. In other word, the stability of a g -adaptive policy is ensured even for $\Delta_m, \Delta_c > 0$. However, it is easy to see that our conclusion regarding the improved performance of dynamic policies over quasi-static policies does not hold when Δ_m, Δ_c increase significantly. As an example, Fig. 11 shows the delay performance under increased monitoring delay $\Delta_m > 0$. In the small Δ_m regime, the AMW policy achieves substantially better performance over the comparing scheduling policies. However, as Δ_m increases, the performance of the AMW policy sees a significant degradation. This observation motivates future research: 1) the development of low-delay ToR monitoring system [19], 2) lower complexity scheduling policies with comparable delay performance, and 3) improvements to the AMW policy in order to increase the robustness with respect to the monitoring and computation delay.

ACKNOWLEDGMENT

This work has been partially supported by L3 Communications, NSF Center for Integrated Access Networks (NSF Grant EEC-0812072), and NSF grant CNS-1329819 Event-Based Information Acquisition, Learning, and Control in High-Dimensional Cyber-Physical Systems.

APPENDIX

The following Foster Lyapunov Theorem is used to show the stability in the proofs listed in this appendix.

Fact 1 (Foster-Lyapunov [20]). *Given a system of edge queues $Q_{ij}, 1 \leq i, j \leq N$, with queue lengths $\mathbf{L}(t) = [L_{ij}(t)]$, which could be described by an irreducible discrete-time Markov chain (DTMC) on a countable state space \mathcal{L} . Let f, g be two nonnegative functions on \mathcal{L} such that $\forall \mathbf{L}(t_k) \in \mathcal{L}$,*

$$\mathbb{E}[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)] \leq -f(\mathbf{L}(t_k)) + g(\mathbf{L}(t_k))$$

and suppose for some $\epsilon > 0$, the set $C = \{\mathbf{L} \in \mathcal{L} : f(\mathbf{L}) < g(\mathbf{L}) + \epsilon\}$ is finite, then the DTMC describing the queue length evolution is positive recurrent and we have

$$\lim_{k \rightarrow \infty} \mathbb{E}[f(\mathbf{L}(t_k))] \leq \lim_{k \rightarrow \infty} \mathbb{E}[g(\mathbf{L}(t_k))]$$

A. Proof of Theorem 1

In the following proof, we use the quadratic Lyapunov function $V(\mathbf{L}) = \langle \mathbf{L}, \mathbf{L} \rangle = \sum_{i,j=1}^N L_{ij}^2$.

Select T such that $T > \frac{\Delta_r}{1-\rho}$, and define the sequence of stopping times as $t_k = kT$. Denote $\Delta(t) = \mathbf{A}(t) - \mathbf{D}(t)$, and we may write the expected drift of the Lyapunov function as

$$\begin{aligned} & \mathbb{E}[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)] \\ &= \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}[\langle \mathbf{L}(t+1), \mathbf{L}(t+1) \rangle - \langle \mathbf{L}(t), \mathbf{L}(t) \rangle | \mathbf{L}(t_k)] \\ &= \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}[\langle 2\mathbf{L}(t), \Delta(t) \rangle + \langle \Delta(t), \Delta(t) \rangle | \mathbf{L}(t_k)] \quad (7) \end{aligned}$$

By the bound on the arrival process in Assumption 1, we have $\Delta_{ij}(t) \leq A_{\max}$, and thus $\langle \Delta(t), \Delta(t) \rangle \leq N^2 A_{\max}^2$.

It now remains to determine the first term in (7). For any queue Q_{ij} , at any time t , the departure $D_{ij}(t)$ differs from $S_{ij}(t)$ only if $L_{ij}(t) = 0$ or it is within the reconfiguration delay. Let \mathbf{R} denote the set of time slots within which the network is doing reconfiguration, then we may write

$$\begin{aligned} \langle \mathbf{L}(t), \mathbf{D}(t) \rangle &= \begin{cases} \langle \mathbf{L}(t), \mathbf{S}(t) \rangle, & \text{if } t \notin \mathbf{R} \\ 0, & \text{if } t \in \mathbf{R} \end{cases} \\ &= \langle \mathbf{L}(t), \mathbf{S}(t) \rangle - \langle \mathbf{L}(t), \mathbf{S}(t) \rangle \mathbf{1}_{\{t \in \mathbf{R}\}} \end{aligned} \quad (8)$$

and thus

$$\begin{aligned} \mathbb{E} [\langle \mathbf{L}(t), \Delta(t) \rangle | \mathbf{L}(t_k)] &= \mathbb{E} [\langle \mathbf{L}(t), \mathbf{A}(t) - \mathbf{D}(t) \rangle | \mathbf{L}(t_k)] \\ &= \langle \mathbf{L}(t), \boldsymbol{\lambda} \rangle - \mathbb{E} [\langle \mathbf{L}(t), \mathbf{S}(t) \rangle | \mathbf{L}(t_k)] \\ &\quad + \mathbb{E} [\langle \mathbf{L}(t), \mathbf{S}(t) \rangle \mathbf{1}_{\{t \in \mathbf{R}\}} | \mathbf{L}(t_k)] \\ &= \mathbb{E} [\langle \mathbf{L}(t), \boldsymbol{\lambda} - \boldsymbol{\Pi}^*(t) \rangle + \langle \mathbf{L}(t), \boldsymbol{\Pi}^*(t) - \mathbf{S}(t) \rangle | \mathbf{L}(t_k)] \\ &\quad + \mathbb{E} [\langle \mathbf{L}(t), \mathbf{S}(t) \rangle \mathbf{1}_{\{t \in \mathbf{R}\}} | \mathbf{L}(t_k)] \end{aligned} \quad (9)$$

where $\boldsymbol{\Pi}^*(t)$ is the MaxWeight schedule at time t .

In [5] the capacity region has been characterized as the convex hull of the feasible schedules, that is

$$\mathcal{C} = \left\{ \sum_{\mathbf{S} \in \mathcal{S}} \alpha_{\mathbf{S}} \mathbf{S} : \sum_{\mathbf{S} \in \mathcal{S}} \alpha_{\mathbf{S}} < 1, \alpha_{\mathbf{S}} \geq 0, \forall \mathbf{S} \in \mathcal{S} \right\}.$$

Hence by the definition of the traffic load ρ , we may write $\boldsymbol{\lambda} = \rho \sum_{i=1}^I \alpha_i \mathbf{S}_i$, where $\mathbf{S}_i \in \mathcal{S}$ and $0 \leq \alpha_i < 1$ for $i = 1, \dots, I$. By the definition of the MaxWeight schedule, we have

$$\begin{aligned} \langle \mathbf{L}(t), \boldsymbol{\lambda} - \boldsymbol{\Pi}^*(t) \rangle &\leq \rho \sum_{i=1}^I \alpha_i W^*(t) - W^*(t) \\ &\leq -(1 - \rho) W^*(t) \end{aligned} \quad (10)$$

Also by the definition of the g -adaptive policy and the fact that the scheduling policy satisfies Condition 1, we have

$$\begin{aligned} \langle \mathbf{L}(t), \boldsymbol{\Pi}^*(t) - \mathbf{S}(t) \rangle &= [W^*(t) - W^\pi(t)] + [W^\pi(t) - W^S(t)] \\ &\leq G + g(W^\pi(t)) \leq G + g(W^*(t)) \end{aligned} \quad (11)$$

Apply (10) and (11) into (9), we obtain

$$\begin{aligned} \mathbb{E} [\langle \mathbf{L}(t), \Delta(t) \rangle | \mathbf{L}(t_k)] &\leq \mathbb{E} [-(1 - \rho) W^*(t) + g(W^*(t)) \\ &\quad + G + W^*(t) \mathbf{1}_{\{t \in \mathbf{R}\}} | \mathbf{L}(t_k)] \end{aligned} \quad (12)$$

In order to achieve the stability, it is necessary to ensure that the reconfiguration does not happen too often. We make this statement formal with the following lemma:

Lemma 1. *Given any fixed $T' > 0$ and a Markov policy π satisfying Condition 1 with weight function $f(x) = x$. Let $g(\cdot)$*

be a sublinear and strictly increasing function, and let $M = g^{-1}(G + N(A_{\max} + 1)T') + NT'$. Suppose a reconfiguration occurs at time t and $W^(t) > M$, then no reconfiguration could occur in $[t + 1, t + T']$.*

The proof for lemma 1 is given in appendix B. Note that lemma 1 gives an upper bound on the frequency of schedule reconfiguration when the queue length is large, since the time between two schedule reconfigurations is larger than T if $W^*(t) > M$. The value of T' in lemma 1 could be arbitrary in general, while in the following we set $T' = T$ (where $T > \frac{\Delta_r}{1-\rho}$ as selected earlier). Thus if $\forall t \in [t_k - \Delta_r, t_{k+1}]$: $W^*(t) > M = g^{-1}(G + N(A_{\max} + 1)T) + NT$, then at most one reconfiguration could occur and we have:

$$\sum_{t=t_k}^{t_{k+1}-1} \mathbf{1}_{\{t \in \mathbf{R}\}} \leq \Delta_r \quad (13)$$

We thus have $\forall \mathbf{L}(t_k) : W^*(t_k) > M$,

$$\begin{aligned} &\mathbb{E} [V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)] \\ &\leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E} [-2(1 - \rho)W^*(t) + 2g(W^*(t)) + 2G | \mathbf{L}(t_k)] \\ &\quad + \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E} [2W^*(t) \mathbf{1}_{\{t \in \mathbf{R}\}} | \mathbf{L}(t_k)] + TN^2 A_{\max}^2 \\ &\leq -2T(1 - \rho)(W^*(t_k) - NT) + 2Tg(W^*(t_k) + NA_{\max}T) \\ &\quad + 2TG + 2\Delta_r(W^*(t_k) + NA_{\max}T) + TN^2 A_{\max}^2 \\ &\leq -\frac{2T}{N}(1 - \rho - \frac{\Delta_r}{T})\|\mathbf{L}(t_k)\| + 2Tg(\|\mathbf{L}(t_k)\| + NA_{\max}T) \\ &\quad + 2T(G + (1 - \rho)NT + NA_{\max}\Delta_r) + TN^2 A_{\max}^2 \end{aligned} \quad (14)$$

where the last inequality follows from $\|\mathbf{L}(t)\| \geq W^*(t) \geq \frac{1}{N}\|\mathbf{L}(t)\|$. Now since $g(\cdot)$ is a sublinear function, there exist constants $B, K < \infty$ and $\epsilon > 0$ such that for $\|\mathbf{L}(t)\| > B$:

$$\mathbb{E} [V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)] \leq -\epsilon\|\mathbf{L}(t_k)\| + K$$

which by Fact 1, implies $\lim_{k \rightarrow \infty} \mathbb{E} \{\|\mathbf{L}(t_k)\|\} \leq K/\epsilon$ and thus the strong stability. Since it holds for any admissible traffic, the g -adaptive variant of π achieves throughput optimality.

Note that (14) gives a bound on the drift for $\|\mathbf{L}(t_k)\| > NM$. On the other hand, for $\|\mathbf{L}(t_k)\| \leq NM$, we also have a simple upper bound on the expected drift as

$$\begin{aligned} &\mathbb{E} [V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) | \mathbf{L}(t_k)] \\ &\leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E} [2\langle \mathbf{L}(t), \Delta(t) \rangle + N^2 A_{\max}^2 | \mathbf{L}(t)] \\ &\leq 2T\|\mathbf{L}(t_k)\|A_{\max} + TN^2 A_{\max}^2 \\ &\leq 2TNMA_{\max} + TN^2 A_{\max}^2 \\ &\leq -\frac{2T}{N}(1 - \rho - \frac{\Delta_r}{T})\|\mathbf{L}(t_k)\| + 2T(1 - \rho - \frac{\Delta_r}{T})M \\ &\quad + 2TNMA_{\max} + TN^2 A_{\max}^2 \end{aligned} \quad (15)$$

Combining (14) and (15) we get a (rather loose) bound on the drift for all $\mathbf{L}(t_k)$ as

$$\begin{aligned} & \mathbb{E} \left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) \middle| \mathbf{L}(t_k) \right] \\ & \leq -\frac{2T}{N} \left(1 - \rho - \frac{\Delta_r}{T} \right) \|\mathbf{L}(t_k)\| + 2Tg \left(\|\mathbf{L}(t_k)\| + NA_{\max}T \right) \\ & \quad + 2T \left(G + N(T + A_{\max}\Delta_r) + M(1 + NA_{\max}) + \frac{N^2 A_{\max}^2}{2} \right) \end{aligned}$$

From Fact 1 we obtain the following bound:

$$\begin{aligned} & \lim_{k \rightarrow \infty} \mathbb{E} \left[\|\mathbf{L}(t_k)\| \right] \\ & \leq \frac{N}{1 - \rho - \frac{\Delta_r}{T}} \left\{ \lim_{k \rightarrow \infty} \mathbb{E} \left[g \left(\|\mathbf{L}(t_k)\| + NA_{\max}T \right) \right] \right. \\ & \quad \left. + \left(G + N(T + A_{\max}\Delta_r) + M(1 + NA_{\max}) + \frac{N^2 A_{\max}^2}{2} \right) \right\} \\ & \leq \frac{N}{1 - \rho - \frac{\Delta_r}{T}} \left\{ g \left(\lim_{k \rightarrow \infty} \mathbb{E} \left[\|\mathbf{L}(t_k)\| \right] + NA_{\max}T \right) \right. \\ & \quad \left. + \left(G + N(T + A_{\max}\Delta_r) + M(1 + NA_{\max}) + \frac{N^2 A_{\max}^2}{2} \right) \right\} \end{aligned}$$

where the last inequality follows from Jensen's inequality given the assumption that $g(\cdot)$ is a concave and continuous function. We then have the bound

$$\lim_{K \rightarrow \infty} \mathbb{E} [\|\mathbf{L}(t)\|] \leq \tilde{L}_T, \quad \text{for each } T > \frac{\Delta_r}{1-\rho} \quad (16)$$

where \tilde{L}_T satisfies $\tilde{L}_T = \frac{N}{1 - \rho - \frac{\Delta_r}{T}} \left\{ g(\tilde{L}_T + NA_{\max}T) + \left(G + N(T + A_{\max}\Delta_r) + M(1 + NA_{\max}) + \frac{N^2 A_{\max}^2}{2} \right) \right\}$.

B. Proof of Lemma 1

Proof: By the assumption, the schedule is reconfigured to $\Pi(t)$ at time t , and according to Condition 1 we have $W^\pi(t) = \langle \mathbf{L}(t), \Pi(t) \rangle \geq W^*(t) - G$. We need to show that at any time $\tau \in [t + 1, t + T]$, the weight of $\Pi(t)$ is large enough so that no schedule reconfiguration occurs.

Since at most one packet could depart at each queue in a time slot, we have

$$\begin{aligned} W(\tau) &= \langle \mathbf{L}(\tau), \Pi(t) \rangle \geq W^\pi(t) - N(\tau - t) \\ &\geq W^*(t) - G - NT \end{aligned} \quad (17)$$

On the other hand, since the arrival at each queue is bounded by A_{\max} , we have an upper bound for the maximum weight:

$$\begin{aligned} W^*(\tau) &= \langle \mathbf{L}(\tau), \Pi^*(\tau) \rangle \leq W^*(t) + NA_{\max}(\tau - t) \\ &\leq W^*(t) + NA_{\max}T \end{aligned} \quad (18)$$

From (17) and (18) we have:

$$W^*(\tau) - W(\tau) \leq G + N(A_{\max} + 1)T$$

Now since the maximum weight is at least the weight of the schedule $\Pi^*(t)$, then from the bound on packet departure as in (17), we have a lower bound for the maximum weight:

$$\begin{aligned} W^*(\tau) &\geq \langle \mathbf{L}(\tau), \Pi^*(t) \rangle \geq W^*(t) - N(\tau - t) \\ &\geq W^*(t) - NT \end{aligned}$$

Hence if $W^*(t) > M$, then $\forall \tau \in [t + 1, t + T]$:

$$\begin{aligned} g(W^*(\tau)) &> g(M - NT) \geq G + N(A_{\max} + 1)T \\ &\geq W^*(\tau) - W(\tau) \end{aligned}$$

Since the weight difference does not exceed the threshold, no schedule reconfiguration could occur within $[t + 1, t + T]$. ■

C. Proof of Theorem 2

Proof: Notice that the policy π (and thus its g -adaptive variant π^g) utilizes randomness in generating schedules, we take this into account when performing the drift analysis. We have the expected drift given by

$$\begin{aligned} & \mathbb{E}^{\pi^g} \left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) \middle| \mathbf{L}(t_k) \right] \\ & \leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g} \left[2 \langle \mathbf{L}(t), \Delta(t) \rangle + N^2 A_{\max}^2 \middle| \mathbf{L}(t_k) \right] \end{aligned} \quad (19)$$

where the expectation is taken with respect to the policy π^g .

Note that (8) in the proof of Theorem 1 remains the same in this case, we thus have

$$\begin{aligned} & \mathbb{E}^{\pi^g} \left[\langle \mathbf{L}(t), \Delta(t) \rangle \middle| \mathbf{L}(t_k) \right] \\ &= \mathbb{E}^{\pi^g} \left[\langle \mathbf{L}(t), \lambda - \Pi^*(t) \rangle + \langle \mathbf{L}(t), \Pi^*(t) - \mathbf{S}(t) \rangle \middle| \mathbf{L}(t_k) \right] \\ & \quad + \mathbb{E}^{\pi^g} \left[\langle \mathbf{L}(t), \mathbf{S}(t) \rangle \mathbb{1}_{\{t \in \mathbf{R}\}} \middle| \mathbf{L}(t_k) \right] \\ &\leq \mathbb{E}^{\pi^g} \left[-(1 - \rho)W^*(t) + \langle \mathbf{L}(t), \Pi^*(t) - \mathbf{S}(t) \rangle \middle| \mathbf{L}(t_k) \right] \\ & \quad + \mathbb{E}^{\pi^g} \left[W^*(t) \mathbb{1}_{\{t \in \mathbf{R}\}} \middle| \mathbf{L}(t_k) \right] \end{aligned} \quad (20)$$

By the construction of the g -adaptive variant and the fact that the policy π satisfies Condition 2, we have for any $t \geq t_k$:

$$\begin{aligned} & \mathbb{E}^{\pi^g} \left[\langle \mathbf{L}(t), \Pi^*(t) - \mathbf{S}(t) \rangle \middle| \mathbf{L}(t_k) \right] \\ &= \mathbb{E}^{\pi^g} \left[\mathbb{E}^\pi \left[W^*(t) - W^\pi(t) \middle| \mathbf{L}(t) \right] \middle| \mathbf{L}(t_k) \right] \\ & \quad + \mathbb{E}^{\pi^g} \left[\mathbb{E}^\pi \left[W^\pi(t) - W^{\pi^g}(t) \middle| \mathbf{L}(t) \right] \middle| \mathbf{L}(t_k) \right] \\ &\leq G + \mathbb{E}^{\pi^g} \left[g(W^*(t)) \middle| \mathbf{L}(t_k) \right] \end{aligned} \quad (21)$$

Similar to the proof in Theorem 1, we need a bound on the rate of schedule reconfiguration. The following lemma works similarly as Lemma 1 except it restricts the rate of reconfiguration in the average sense.

Lemma 2. *Given any fixed $T' > 0$ and a scheduling policy π satisfying Condition 1 under the weight function $f(x) = x$.*

Let $g(x)$ be a sublinear and strictly increasing function, and let $h(x) = g(x - NT) - NA_{\max}T$. Then

$$\mathbb{E}^{\pi^g} \left[\sum_{t=t_k}^{t_{k+1}-1} \mathbf{1}_{\{t \in \mathbf{R}\}} \middle| \mathbf{L}(t_k) \right] \leq \Delta_r + \frac{TG}{h(W^*(t_k))}$$

The proof of Lemma 2 is given in Appendix D and follows the similar approach as in Lemma 1.

Since $W^*(t_k) - NT \leq W^*(t) \leq W^*(t_k) + NA_{\max}T$ for all $t \in [t_k, t_{k+1}]$, applying Lemma 2 we obtain

$$\begin{aligned} & \mathbb{E}^{\pi^g} \left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) \middle| \mathbf{L}(t_k) \right] \\ & \leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g} \left[2 \langle \mathbf{L}(t), \Delta(t) \rangle + N^2 A_{\max}^2 \middle| \mathbf{L}(t_k) \right] \\ & \leq \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g} \left[-2(1-\rho)W^*(t) + 2g(W^*(t)) + 2G \middle| \mathbf{L}(t_k) \right] \\ & \quad + \sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}^{\pi^g} \left[2W^*(t) \mathbf{1}_{\{t \in \mathbf{R}\}} + N^2 A_{\max}^2 \middle| \mathbf{L}(t_k) \right] \\ & \leq -2T \left(1 - \rho - \frac{\Delta_r}{T} - \frac{G}{h(W^*(t_k))} \right) W^*(t_k) \\ & \quad + 2Tg(W^*(t_k) + NA_{\max}T) + TN^2 A_{\max}^2 \\ & \quad + 2T \left(G + N(T + A_{\max}\Delta_r) + \frac{NA_{\max}GT}{h(W^*(t_k))} + \frac{N^2 A_{\max}^2}{2} \right) \end{aligned}$$

Since $T > \frac{\Delta_r}{1-\rho}$, we have $1 - \rho - \frac{\Delta_r}{T} > 0$, and we may select an arbitrary constant $\epsilon \in (0, 1 - \rho - \frac{\Delta_r}{T})$. Now since $\lim_{x \rightarrow \infty} h(x) = \infty$, there exists a constant $M_1 < \infty$ such that $W^*(t_k) > M_1$ implies $1 - \rho - \frac{\Delta_r}{T} - \frac{G}{h(W^*(t_k))} > \epsilon$. Also, by the sublinearity of $g(\cdot)$, there exists a constant M_2 such that $W^*(t_k) > M_2$ implies $g(W^*(t_k)) < \frac{\epsilon}{3}W^*(t_k)$. Now let $M_3 = \frac{3}{\epsilon} \left(G + N(T + A_{\max}\Delta_r) + \frac{NA_{\max}GT}{h(W^*(t_k))} + \frac{N^2 A_{\max}^2}{2} \right)$ and $B = \max\{M_1, M_2, M_3\}$, we have that if $W^*(t_k) > B$, then

$$\begin{aligned} & \mathbb{E}^{\pi^g} \left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) \middle| \mathbf{L}(t_k) \right] \\ & < -\epsilon W^*(t_k) + \frac{\epsilon}{3}W^*(t_k) + \frac{\epsilon}{3}W^*(t_k) = -\frac{\epsilon}{3}W^*(t_k) \end{aligned}$$

Recall that $W^*(t) \geq \frac{1}{N} \|\mathbf{L}(t)\|$ at any time t , we have that for $\|\mathbf{L}(t_k)\| > NB$:

$$\mathbb{E}^{\pi^g} \left[V(\mathbf{L}(t_{k+1})) - V(\mathbf{L}(t_k)) \middle| \mathbf{L}(t_k) \right] < -\frac{\epsilon}{3N} \|\mathbf{L}(t_k)\| \quad (22)$$

then from Fact 1 we have that the network is strongly stable under any feasible traffic load, hence the g -adaptive variant of π achieves throughput optimality. \blacksquare

D. Proof of Lemma 2

Proof: Let $Z_{[s,t]}$ be the number of schedule reconfigurations within the time period $[s, t]$. Recall in lemma 1 we showed that $Z_{[t_k, t_{k+1}]} \leq 1$ given $W^*(t_k)$ is large. Here we show a similar result in the probabilistic sense, in particular, we would derive a bound on $\Pr\{Z_{[t_k, t_{k+1}]} \leq 1\}$ in the following.

Let $t'_k = \min_{k: t_k^S \geq t} t_k^S$ denote the first reconfiguration instance after time t_k . We may then write the event that exactly one schedule reconfiguration occurs in the interval $[t_k, t_{k+1}]$ as:

$$\{Z_{[t_k, t_{k+1}]} = 1\} = \bigcup_{\tau=t_k}^{t_{k+1}-1} \{\{\tau = t'_k\} \cap \{Z_{[\tau, t_{k+1}]} = 0\}\}.$$

Let $E_\tau = \{\{\tau = t'_k\} \cap \{Z_{[\tau, t_{k+1}]} = 0\}\}$. Note E_τ is the event that there is exactly one schedule reconfiguration within $[t_k, t_{k+1}]$ and it occurs at time τ . The events E_τ are disjoint events for different τ , hence we have that

$$\begin{aligned} \Pr\{Z_{[t_k, t_{k+1}]} = 1 \middle| \mathbf{L}(t)\} &= \sum_{\tau=t_k}^{t_{k+1}-1} \Pr\{E_\tau \middle| \mathbf{L}(t)\} \\ &= \sum_{\tau=t_k}^{t_{k+1}-1} \Pr\{Z_{[\tau, t_{k+1}]} = 0 \middle| \mathbf{L}(t), \tau = t'_k\} \Pr\{\tau = t'_k \middle| \mathbf{L}(t)\} \end{aligned} \quad (23)$$

Following the similar approach in Lemma 1, we show that if at the schedule reconfiguration instance τ , the weight difference is small, then no reconfiguration could occur in the interval $[\tau, t_{k+1}]$. Specifically, let $h(W^*(t_k)) = g(W^*(t_k) - NT) - N(A_{\max} + 1)T$, then if $W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k))$, we have for any $\tau' \in [\tau, t_{k+1}]$:

$$\begin{aligned} & W^*(\tau') - W_{\Pi^g(\tau)}(\tau') \\ & \leq W^*(\tau) - W_{\Pi^g(\tau)}(\tau) + N(A_{\max} + 1)(\tau' - \tau) \\ & < g(W^*(t_k) - NT) \stackrel{(a)}{\leq} g(W^*(\tau')) \end{aligned}$$

where (a) follows from $W^*(\tau') \geq W^*(t) - N(\tau' - t) \geq W^*(t) - NT$ and g strictly increasing. We thus have:

$$\Pr\{Z_{[\tau, t_{k+1}]} = 0 \middle| W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k))\} = 1.$$

We then obtain:

$$\begin{aligned} & \Pr\{Z_{[\tau, t_{k+1}]} = 0 \middle| \tau = t'_k\} \\ &= \Pr\{Z_{[\tau, t_{k+1}]} = 0 \middle| W^*(\tau) - W_{\Pi^g(\tau)}(\tau) \geq h(W^*(t_k))\} \\ & \quad \Pr\{W^*(\tau) - W_{\Pi^g(\tau)}(\tau) \geq h(W^*(t_k)) \middle| \tau = t'_k\} \\ & \quad + \Pr\{Z_{[\tau, t_{k+1}]} = 0 \middle| W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k))\} \\ & \quad \Pr\{W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k)) \middle| \tau = t'_k\} \\ & \geq \Pr\{W^*(\tau) - W_{\Pi^g(\tau)}(\tau) < h(W^*(t_k)) \middle| \tau = t'_k\} \\ & \stackrel{(b)}{\geq} 1 - \frac{\mathbb{E}^\pi [W^*(\tau) - W_{\Pi^g(\tau)}(\tau)]}{h(W^*(t_k))} \stackrel{(c)}{\geq} 1 - \frac{G}{h(W^*(t_k))} \end{aligned} \quad (24)$$

where (b) follows from the conditional Markov's inequality, and (c) follows from Condition 2.

Notice that either there is no reconfigurations within $[t_k, t_{k+1}]$ or otherwise the first reconfiguration occurs at some $\tau \in [t_k, t_{k+1}]$, we thus have

$$\Pr \left\{ Z_{[t_k, t_{k+1}]} = 0 \middle| \mathbf{L}(t) \right\} + \sum_{\tau=t_k}^{t_{k+1}-1} \Pr \left\{ \tau = t'_k \middle| \mathbf{L}(t) \right\} = 1$$

hence by (23) and (24) we have that

$$\begin{aligned} & \Pr \left\{ Z_{[t_k, t_{k+1}]} \leq 1 \middle| \mathbf{L}(t) \right\} \\ = & \Pr \left\{ Z_{[t_k, t_{k+1}]} = 0 \middle| \mathbf{L}(t) \right\} \\ & + \sum_{\tau=t_k}^{t_{k+1}-1} \Pr \left\{ Z_{[\tau, t_{k+1}]} = 0 \middle| \mathbf{L}(t), \tau = t'_k \right\} \Pr \left\{ \tau = t'_k \middle| \mathbf{L}(t) \right\} \\ \geq & \Pr \left\{ Z_{[t_k, t_{k+1}]} = 0 \middle| \mathbf{L}(t) \right\} \\ & + \sum_{\tau=t_k}^{t_{k+1}-1} \left(1 - \frac{G}{h(W^*(t))} \right) \Pr \left\{ \tau = t'_k \middle| \mathbf{L}(t) \right\} \\ \geq & 1 - \frac{G}{h(W^*(t))} \end{aligned} \quad (25)$$

With the following bound which is obvious by definition:

$$\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{t \in \mathbf{R}\}} \leq \begin{cases} \Delta_r, & \text{if } Z_{[t_k, t_{k+1}]} \leq 1 \\ T, & \text{if } Z_{[t_k, t_{k+1}]} > 1 \end{cases},$$

along with (25), we then have the bound on the expected schedule reconfiguration delay within the interval $[t_k, t_{k+1}]$:

$$\mathbb{E}^{\pi^g} \left[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{1}_{\{t \in \mathbf{R}\}} \middle| \mathbf{L}(t_k) \right] \leq \Delta_r + \frac{TG}{h(W^*(t_k))}$$

■

REFERENCES

- [1] "Nistica Wavelength Selective Switch Product Data Sheet." <http://www.nistica.com/products.html>.
- [2] B. Keyworth, "Roadm subsystems and technologies," in *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, vol. 3, pp. 4 pp. Vol. 3-, March 2005.
- [3] K. Ross and N. Bambos, "Adaptive batch scheduling for packet switching with delays," in *High-performance Packet Switching Architectures* (I. Elhanany and M. Hamdi, eds.), pp. 65–79, Springer London, 2007.
- [4] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Ros-ing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 447–458, ACM, 2013.
- [5] G. Celik and E. Modiano, "Scheduling in networks with time-varying channels and reconfiguration delay," *Networking, IEEE/ACM Transactions on*, vol. 23, pp. 99–113, Feb 2015.
- [6] Y. Li, S. Panwar, and H. Chao, "Frame-based matching algorithms for optical switches," in *High Performance Switching and Routing, 2003. HPSR. Workshop on*, pp. 97–102, June 2003.
- [7] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, pp. 1936–1948, Dec 1992.
- [8] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 533–539 vol.2, Mar 1998.
- [9] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *Selected Areas in Communications, IEEE Journal on*, vol. 21, pp. 546–559, May 2003.
- [10] D. Shah and D. Wischik, "Optimal scheduling algorithms for input-queued switches," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–11, April 2006.
- [11] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *INFOCOM, 2015 Proceedings IEEE*, April 2015.
- [12] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 1024–1031 vol.2, 2002.
- [13] I. Keslassy, R. Zhang-Shen, and N. McKeown, "Maximum size matching is unstable for any packet switch," *Communications Letters, IEEE*, vol. 7, pp. 496–498, Oct 2003.
- [14] I. Keslassy and N. McKeown, "Analysis of scheduling algorithms that provide 100% throughput in input-queued switches," 2001.
- [15] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1614–1623 vol.3, Mar 2000.
- [16] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A5*, no. 147-150, 1946.
- [17] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI'14*, (Berkeley, CA, USA), pp. 1–15, USENIX Association, 2014.
- [18] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A Centralized Zero-Queue Datacenter Network," in *ACM SIGCOMM 2014*, (Chicago, IL), August 2014.
- [19] T. Aktas, C.-H. Wang, and T. Javidi, "Wicod: Wireless control plane serving an all-optical data center," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2015 13th International Symposium on, pp. 299–306, May 2015.
- [20] B. Hajek, "Notes for ece 534: An exploration of random processes for engineers," 2009.